# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. ...................................................................................09/902,812
Filing Date ............................................................................................ 7/10/2001
Inventorship............................................................................... Hejlsberg et al.
Applicant........................................................................................Microsoft Corp.
Group Art Unit ...............................................................................................2194
Examiner ..................................................................................... Cao, Diem K.
Attorney's Docket No. ...................................................................... ms1-866us
Title: "Application Program Interface for Network Software Platform "

## SUPPLEMENTAL APPEAL BRIEF

To:     Commissioner for Patents
        PO Box 1450
        Alexandria, Virginia  22313-1450


From:   Rich Bucher (Tel. 509-324-9256x216; Fax 509-323-8979)
        **Customer No. 22801**


Pursuant to 37 C.F.R. §41.37, Applicant hereby submits an appeal brief

for application 09/902,812, filed July 10, 2001, within the requisite time from the

date of filing the Notice of Appeal.  Accordingly, Applicant appeals to the Board of

Patent Appeals and Interferences seeking review of the Examiner's rejections.

| **Appeal Brief Items** | **Page** |
|---|---|

## (1)    Real Party in Interest

The real party in interest is Microsoft Corporation, the assignee of all right, title and interest in and to the subject invention.

## (2)    Related Appeals and Interferences

Appellant is not aware of any other appeals, interferences, or judicial proceedings which will directly affect, be directly affected by, or otherwise have a bearing on the Board's decision to this pending appeal.

## (3)    Status of Claims

Claims 1, 3-16, and 18-40 stand rejected and are pending in the Application. Claims 1, 3-16, and 18-40 are set forth in the Appendix of Appealed Claims on Page 30.

Claims 2 and 17 were cancelled without prejudice.

## (4)    Status of Amendments

**The most recent Final Office Action** was mailed April 4, 2006. No amendments were made thereafter.

An interview with the Examiner was conducted by telephone on April 21, 2006.

A Notice of Appeal and Request for Reconsideration were filed concurrently on June 23, 2006.

Application No. 09/902,812

## (5)    Summary of Claimed Subject Matter

A concise explanation of each of the independent claims is included in this Summary section, including specific reference characters, if any. These specific reference characters are examples of particular elements of the drawings for certain embodiments of the claimed subject matter and the claims are not limited to solely the elements corresponding to these reference characters.

With regard to claim 1, a software architecture (Fig. 2) for a distributed computing system comprising: an application (Fig. 1 (110), Fig. 2 (130), Page 7, line 24 through Page 8, line 6) configured to handle requests submitted by remote devices (Fig. 4 (448, 458) over a network (Page 9, lines 3-18); and an application program interface (Fig. 1 (142), Fig. 2 (142)) to present functions used by the application to access network and computing resources of the distributed computing system (Page 9, lines 3-13), the application program interface comprising various types related to constructing user interfaces, wherein the various types comprise: classes which represent managed heap allocated data that has reference assignment semantics; interfaces that define a contract that other types can implement; delegates that are object oriented function pointers; structures that represent static allocated data that has value assignment semantics; and enumerations which are value types that represent named constants. (Page 11, line 13 through Page 13, line 8).

4

With regard to claim 5, An application program interface (Fig. 1 (142), Fig. 2 (142)) embodied on one or more computer readable media (Fig. 4 (406) Page 2975, line 15) comprising: multiple types related to constructing user interfaces, the types comprising classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, structures that represent static allocated data that has value assignment semantics and enumerations which are value types that represent named constants (Page 11, line 13 through Page 13, line 8).

With regard to claim 16, a distributed computer software architecture, comprising: one or more applications (Fig. 2 (130), Page 7, line 24 through Page 8, line 6) configured to be executed on one or more computing devices, the applications handling requests submitted from remote computing devices (Fig. 4 (448, 458)); a networking platform (Page 3, lines 15-20, Fig. 2) to support the one or more applications; and an application programming interface (Fig. 1 (142), Fig. 2 (142)) to interface the one or more applications with the networking platform (Page 9, lines 3-13), the application programming interface comprising various types related to constructing user interfaces, wherein the various types comprise: classes which represent managed heap allocated data that has reference assignment semantics; interfaces that define a contract that other types can implement; delegates that are object oriented function pointers; structures that represent static allocated data that has value assignment semantics; and enumerations which are value types that represent named constants. (Page 11, line 13 through Page 13, line 8).

5

With regard to claim 28, a computer system (Fig. 4) including one or more microprocessors (Fig. 4 (404)) and one or more software programs (Fig. 4 (428), Fig. 1 (130), Fig. 2 (130)), the one or more software programs utilizing an application program interface (Fig. 1 (142), Fig. 2 (142)) to request services from an operating system (Fig. 1 (146), Fig. 2 (146)), the application program interface including separate commands to request services comprising services related to constructing user interfaces (Page 11, lines 3-13), wherein the application program interface groups API functions into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics (Page 11, lines 13-20).

With regard to claim 29, a method, comprising: managing network and computing resources for a distributed computing system (Page 9, lines 3-13); and exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system (Page 9, lines 3-13), the set of functions comprising functions to facilitate construction of user interfaces (Page 11, lines 3-13), wherein the functions are grouped into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures

6

that represent static allocated data that has value assignment semantics (Page 11, lines 13-20).

Regarding claim 31, a method, comprising creating a namespace with functions that enable drawing and construction of user interfaces, the name space defining classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, structures that represent static allocated data that has value assignment semantics, and enumerations which are value types that represent named constants (Page 11, lines 3-20, Page 12, lines 15-21, Pages 13, line 10 through Page 22, line 5, Fig. 2 (200)).

### (6)    Grounds of Rejection to be Reviewed on Appeal

Claims 31-40 Claims stand rejected under 35 U.S.C. § 101 as allegedly being directed to non-statutory subject matter.

Claims 1, 3-16, 18-40 stand rejected under 35 U.S.C. § 103(a) over a publication by Cohn et al. (hereinafter "Cohn") in view of a publication by Flanagan (hereinafter "Flanagan") and further in view of a publication by Microsoft (hereinafter "Microsoft").

### (7)    Arguments
**A.    The rejection under 35 U.S.C. § 101 is inappropriate and should be withdrawn.**

Claims 31-40 stand rejected under 35 U.S.C. § 101 as allegedly being directed to non-statutory subject matter. In making out the rejection of these claims, the Office argues that:

The claimed invention as a whole must be useful and accomplish a practical application, that is, it must produce a "useful, concrete and tangible result". Claims 31-40 are directed to a method comprising creating a namespace with functions that enable drawing and construction of interfaces, ***thus creating a namespace does not provide a useful, concrete and tangible result***, thus the claims claim non-statutory subject matter. (Office Action, mailed April 4, 2006, Page 2) (emphasis added).

Applicant respectfully disagrees with the Office's assertion and traverses this rejection.

It is established law that an abstract idea, by itself, is considered to be unpatentable subject matter under § 101. See, e.g., AT&T Corp. v. Excel Communications, Inc., 172 F.3d 1352, 1355 (1999)(pointing out that laws of nature, natural phenomena, and abstract ideas have generally been identified by the Supreme Court as unpatentable subject matter). However, if such an idea is taken out of the abstract and employed in some type of process that achieves a "new and useful end", the *process is* patentable subject matter, even if the idea by itself would not be. Id. at 1357. Thus, the relevant inquiry under § 101 becomes -- Is the idea being applied to achieve a useful end? Id. If so, then the § 101 threshold is satisfied. Id.

In AT&T, the invention was designed to operate in a telecommunications system with multiple long-distance service providers. The system contains local exchange carriers ("LECs") and long-distance service (interexchange) carriers ("IXCs"). The LECs provide local telephone service and access to IXCs. Each customer has an LEC for local service and selects an IXC, such as AT & T or Excel, to be its primary long-distance service (interexchange) carrier or PIC. The

Application No. 09/902,812

system involves a three-step process when a caller makes a direct-dialed (1+) long-distance telephone call: (1) after the call is transmitted over the LEC's network to a switch, and the LEC identifies the caller's PIC, the LEC automatically routes the call to the facilities used by the caller's PIC; (2) the PIC's facilities carry the call to the LEC serving the call recipient; and (3) the call recipient's LEC delivers the call over its local network to the recipient's telephone.

When a caller makes a direct-dialed long-distance telephone call, a switch (which may be a switch in the interexchange network) monitors and records data related to the call, generating an "automatic message account" ("AMA") message record. This contemporaneous message record contains fields of information such as the originating and terminating telephone numbers, and the length of time of the call. These message records are then transmitted from the switch to a message accumulation system for processing and billing.

Because the message records are stored in electronic format, they can be transmitted from one computer system to another and reformatted to ease processing of the information. Thus the carrier's AMA message subsequently is translated into the industry-standard "exchange message interface," forwarded to a rating system, and ultimately forwarded to a billing system in which the data resides until processed to generate, typically, "hard copy" bills which are mailed to subscribers.

The invention at issue in this case called for the addition of a data field into a standard message record to indicate whether a call involves a particular PIC (the "PIC indicator"). This PIC indicator can exist in several forms, such as a code which identifies the call recipient's PIC, a flag which shows that the recipient's

PIC is or is not a particular IXC, or a flag that identifies the recipient's and the caller's PICs as the same IXC. The PIC indicator therefore enables IXCs to provide differential billing for calls on the basis of the identified PIC.

One of the claims at issue – claim 1-- read as follows:

> A method for use in a telecommunications system in which interexchange calls initiated by each subscriber are automatically routed over the facilities of a particular one of a plurality of interexchange carriers associated with that subscriber, said method comprising the steps of:
> generating a message record for an interexchange call between an originating subscriber and a terminating subscriber, and
> including, in said message record, a primary interexchange carrier (PIC) indicator having a value which is a function of whether or not the interexchange carrier associated with said terminating subscriber is a predetermined one of said interexchange carriers.

In looking at the subject matter of this claim and finding the claim to pass muster under 35 U.S.C. § 101, the Court looked to the *specification* and commented as follows:

> In this case, Excel argues, correctly, that the PIC indicator value is derived using a simple mathematical principle (p and q). But that is not determinative because AT&T does not claim the Boolean principle as such or attempt to forestall its use in any other application. It is clear from the written description of the '184 patent that AT&T is only claiming a process that uses the Boolean principle in order to determine the value of the PIC indicator. The PIC indicator represents information about the call recipient's PIC, a useful, non-abstract result that facilitates differential billing of long-distance calls made by an IXC's subscriber. Because the claimed process applies the Boolean principle to produce a useful, concrete, tangible result without pre-empting other uses of the mathematical principle, on its face the claimed process comfortably falls within the scope of § 101.

10

Here, the Court looked at the specification and found that the environment and use of the PIC indicator – that of providing differential billing – *provided a useful, concrete and tangible result.  That result, however, was not specifically recited in the claim.  Rather, it was described in the specification*. (emphasis added).

Likewise, in the present case, *the specification provides a description of the utility and tangibility of "creating a namespace", as claimed*.  By way of example and not limitation, Applicant directs the Office's attention to two excerpts from the specification on Pages 12 (lines 15-21) and 14 (line 24) through 15 (line 15), respectively:

The Web applications namespace 200 pertains to Web based functionality, such as dynamically generated Web Pages (e.g., Microsoft's Active Server Pages (ASP)).  It supplies types that enable browser/server communication.  The client applications namespace 202 pertains to drawing and client side UI functionality.  It supplies types that enable drawing of two-dimensional (2D), imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on.

A UI namespace 312 ("System.Web.UI") containing types that allow developers to create controls and Pages that will appear in Web applications as user interfaces on a Web Page.  This namespace includes the control class, which provides all web based controls, whether those encapsulating HTML elements, higher level Web controls, or even custom User controls, with a common set of functionality. Also provided are classes which provide the web forms server controls data binding functionality, the ability to save the view state of a given control or Page, as well as parsing functionality for both programmable and literal controls.  Within the UI namespace 312 are two additional namespaces: an HTML controls namespace 314 ("System.Web.UI.HtmlControls") containing classes that permit developers to interact with types that encapsulates html 3.2 elements create HTML controls, and a Web controls

namespace 316 ("System.Web.UI.WeblControls") containing classes that allow developers to create higher level Web controls.

Accordingly, in these excerpts, as throughout the document, it is evident that the claimed subject matter has a specifically described useful, concrete and tangible result and application.

In view of the above discussion, the Office has failed to show that claims 31-40 present unpatentable subject matter under § 101. Applicant respectfully submits that claims 31-40 comply with the patentability requirements of § 101 and that the rejections should be withdrawn.

**B.      The rejections under 35 U.S.C. §103(a) over Cohn, Flanagan Microsoft fail because the Office has failed to establish a *prima facie* case of obviousness.**

Applicant respectfully submits that the Office has not established a *prima facie* case of obviousness. The discussion below proceeds as follows. First, a section entitled "The § 103 Standard" is provided which describes the criteria that must be met in order to establish a *prima facie* case of obviousness. Second, a section entitled "The Claims" is provided which presents Applicant's reasoning as to why the Office has not met these criteria.

Application No. 09/902,812

## The §103 Standard

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, **there must be some suggestion or motivation**, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, **to modify the reference or to combine reference teachings**. Second, there must be a reasonable expectation of success. Finally, **the prior art reference (or references when combined) must teach or suggest all the claim limitations**. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on Applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

## The Claims

**Claim 1** recites a software architecture for a distributed computing system comprising:

- an application configured to handle requests submitted by remote devices over a network; and
- an application program interface to present functions used by the application to access network and computing resources of the distributed computing system, the application program interface comprising various types related to constructing user interfaces, wherein the various types comprise:
  - classes which represent managed heap allocated data that has reference assignment semantics;
  - interfaces that define a contract that other types can implement;
  - delegates that are object oriented function pointers;
  - structures that represent static allocated data that has value assignment semantics; and
  - enumerations which are value types that represent named constants.

In making out the rejection of this claim, the Office first argues that Chapter 19 of Cohn teaches an application and Chapter 17 teaches an "application interface" (API), as claimed, "comprising various types related to constructing user interfaces". However, the Office then argues that Chapters 5 and 6 teach "an API comprising multiple types related to construction user interfaces" and "classes which represent managed heap allocated data that has reference assignment semantics". The Office then relies on Flanagan for disclosing "interfaces", "structures" and "enumerations" and on Microsoft for disclosing "delegates". The Office states that it would have been obvious to integrate the teachings of these references "because Visual J++ is based on Java, and Flanagan teaches details of classes and packages offered by Java, and Microsoft teaches delegates address many scenarios that are addressed by function pointers..."

Applicant respectfully disagrees and traverses the Office's rejection. First, Applicant submits that it remains unclear what teachings of Cohn the Office is relying on for disclosing "an application program interface" (API), as claimed. Specifically, on Page 3 of the Office Action, the Office argues that Chapters 5 and 6 teaches an API, as claimed. Then, on Page 6 of the Office Action, it inexplicably argues that Chapter 19 teaches an API, as claimed.

Nevertheless, Applicant has thoroughly reviewed Chapters 5, 6, and 19 and is unable to find any discussion of an API, as recited in this claim. Unfortunately, the Office offers no explanation other than to state: "[i]n Java, and applets....Scrollbars; chapter 5, Page 1 and Container class; chapter 6, Page 1 and through out chapters 5 and 6". Applicant fails to see how this statement is

pertinent and submits that these excerpts simply do not teach the API recited in this claim. Instead, Chapter 5 describes how to use Java components with respect to creating a user interface and Chapter 6 describes how to combine and position these components. Furthermore, Chapter 17 merely discusses the nature of URLs and how they can be used. Missing from these excerpts, and from Cohn in general, is any discussion of an API "comprising various types related to constructing user interfaces", as claimed. This is not surprising because Cohn is described as a book that "tells how to use the language to solve problems" and states: "[t]his book is not an API reference". (See Introduction, Page 12).

Additionally, Applicant respectfully submits that the Office is interpreting the term "application program interface" in a manner that is inconstant with how it is understood and used in the subject application. Applicant respectfully reminds the Office that claims are not to be considered in a vacuum, but rather in the context of the specification and drawings. (see e.g., MPEP 2106(II)(c), 2111(I)). In this regard, Applicant directs the Office's attention to Page 6 (lines 8-10) of the subject application, which is one example of how this term is understood and used in the context of the subject application. Even a cursory inspection of Page 6 (lines 8-10) is sufficient to distinguish this claim from any teachings found in the cited excerpts of Cohn. This excerpt is reproduced below for the Office's convenience:

> As used herein, the phrase application program interface or API includes traditional interfaces that employ method or function calls, as well as remote calls (e.g., a proxy, stub relationship) and SOAP/XML invocations.

Furthermore, Applicant respectfully submits that the cited excerpts of Flanagan neither disclose nor suggest an API comprising all of the various types, as claimed. For instance, Applicant fails to see how Fig. 19-1 and Page 238 of Flanagan disclose "interfaces, that define a contract that other types can implement", as claimed. Instead, this figure and excerpt merely discuss *classes* of the java.awt package.

Finally, and perhaps most importantly, Applicant respectfully reminds the Office that to support a conclusion that the claimed subject matter is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed subject matter or the examiner must present a convincing line of reasoning as to *why* an artisan would have found the claimed subject matter to have been obvious in light of the teachings of the references. (see MPEP 2142 and 2143.01).

Here, the Office's only attempt at such an explanation is to state that it would have been obvious to combine the teachings of these references "because Visual J++ is based on Java, and Flanagan teaches details of classes and packages offered by Java". However, upon close examination, this statement merely describes the references themselves and fails to explain *why* one would have been motivated to combine their teachings. As such, this statement actually fails to articulate any motivation at all. Accordingly, as best as Applicant can discern, the Office's argument relies solely on the fact that the references *can* be combined. However, whether or not the references *can* be combined is irrelevant because: "the mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of

16                                    Application No. 09/902,812

the combination." (MPEP 2143.01, citing *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed Cir. 1990)).

Further, even if the cited references did disclose or suggest all the claimed subject matter, which they do not, it remains unclear why one would have been motivated to combine them in a manner similar to this claim. As noted above, the Office has not provided any explanation, or even a stated motivation, in this regard. Accordingly, Applicant can only conclude that the Office has impermissibly used the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious. Applicant respectfully reminds the Office that "[o]ne cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention." (quoting *In Re Fine*, 837 F.2d 1071, 1075, 5 USPQ2d 1596, 1600 (Fed. Cir. 1988)).

Accordingly, in view of the above discussion, the Office has not established a *prima facie* case of obviousness. Hence, for at least these reasons, this claim is allowable.

**Claims 3-4** depend from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

**Claim 5** recites an application program interface embodied on one or more computer readable media, comprising: multiple types related to constructing user interfaces, the types comprising classes which represent managed heap allocated

data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, structures that represent static allocated data that has value assignment semantics and enumerations which are value types that represent named constants.

In making out the rejection of this claim, the Office relies on the same argument that it made with respect to claim 1 except that it does not rely on Chapter 17 as teaching "an application program interface", as claimed.

Applicant respectfully disagrees and traverses the Office's rejection. First, as discussed above, Applicant submits that it is unclear what teachings of Cohn the Office is relying on for disclosing "an application program interface" (API), as claimed. Nevertheless, Applicant has thoroughly reviewed Chapters 5, 6, and 19 and is unable to find any discussion of an API, as recited in this claim. Instead, Chapter 5 describes how to use Java components with respect to creating a user interface and Chapter 6 describes how to combine and position these components. Furthermore, Chapter 17 merely discusses the nature of URLs and how they can be used. Missing from these excerpts, and from Cohn in general, is any discussion of "an application program interface ... comprising: multiple types related to constructing user interfaces", as claimed.

Additionally, Applicant respectfully submits that the Office is interpreting the term "application program interface" in a manner that is inconstant with how it is understood and used in the subject application. In this regard, Applicant directs the Office's attention to Page 6 (lines 8-10) of the subject application, which is one example of how this term is understood and used in the context of the subject

application. Even a cursory inspection of Page 6 (lines 8-10) is sufficient to distinguish this claim from any teachings found in the cited excerpts of Cohn.

Furthermore, Applicant respectfully submits that the cited excerpts of Flanagan neither disclose nor suggest an API comprising all of the various types, as claimed. For instance, Applicant fails to see how Fig. 19-1 and Page 238 of Flanagan disclose "interfaces that define a contract that other types can implement", as claimed. Instead, this figure and excerpt merely discuss *classes* of the java.awt package.

Finally, and perhaps most importantly, as noted above, the Office's stated motivation merely describes the references themselves and fails to explain *why* one would have been motivated to combine their teachings. As such, this statement actually fails to articulate any motivation at all. Accordingly, as best as Applicant can discern, the Office's argument relies solely on the fact that the references *can* be combined. However, whether or not the references *can* be combined is irrelevant. Further, even if the cited references did disclose or suggest all the claimed subject matter, which they do not, it remains unclear why one would have been motivated to combine them in a manner similar to this claim. Applicant can only conclude that the Office has impermissibly used the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious.

Accordingly, in view of the above discussion, the Office has not established a *prima facie* case of obviousness. Hence, for at least these reasons, this claim is allowable.

**Claims 6-15** depend from claim 5 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 5, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

Additionally, regarding claims 8-15, Applicant submits that the excerpts from Cohn that are cited by the Office disclose various *classes*, not interfaces.

Accordingly, the Office's reliance on these excerpts is misplaced.

**Claim 16** recites a distributed computer software architecture comprising:

- one or more applications configured to be executed on one or more computing devices, the applications handling requests submitted from remote computing devices;
- a networking platform to support the one or more applications; and
- an application programming interface to interface the one or more applications with the networking platform, the application programming interface comprising various types related to constructing user interfaces, wherein the various types comprise:
- classes which represent managed heap allocated data that has reference assignment semantics;
- interfaces that define a contract that other types can implement;
- delegates that are object oriented function pointers;
- structures that represent static allocated data that has value assignment semantics; and
- enumerations which are value types that represent named constants.

In making out the rejection of this claim, the Office relies on the same argument that it made with respect to claim 1.

Applicant respectfully disagrees and traverses the Office's rejection. First, as discussed above, Applicant submits that it is unclear what teachings of Cohn the Office is relying on for disclosing "an application programming interface" (API),

as claimed. Nevertheless, Applicant has thoroughly reviewed Chapters 5, 6, and 19 and is unable to find any discussion of an API, as recited in this claim. Instead, Chapter 5 describes how to use Java components with respect to creating a user interface and Chapter 6 describes how to combine and position these components. Furthermore, Chapter 17 merely discusses the nature of URLs and how they can be used. Missing from these excerpts, and from Cohn in general, is any discussion of an "application programming interface comprising various types related to constructing user interfaces", as claimed.

Additionally, Applicant respectfully submits that the Office is interpreting the term "application programming interface" in a manner that is inconstant with how it is understood and used in the subject application. In this regard, Applicant directs the Office's attention to Page 6 (lines 8-10) of the subject application, which is one example of how this term is understood and used in the context of the subject application. Even a cursory inspection of Page 6 (lines 8-10) is sufficient to distinguish this claim from any teachings found in the cited excerpts of Cohn.

Furthermore, Applicant respectfully submits that the cited excerpts of Flanagan neither disclose nor suggest an API comprising all of the various types, as claimed. For instance, Applicant fails to see how Fig. 19-1 and Page 238 of Flanagan disclose "interfaces that define a contract that other types can implement", as claimed. Instead, this figure and excerpt merely discuss *classes* of the java.awt package.

Finally, and perhaps most importantly, as noted above, the Office's stated motivation merely describes the references themselves and fails to explain *why* one would have been motivated to combine their teachings. As such, this

statement actually fails to articulate any motivation at all. Accordingly, as best as Applicant can discern, the Office's argument relies solely on the fact that the references *can* be combined. However, whether or not the references *can* be combined is irrelevant. Further, even if the cited references did disclose or suggest all the claimed subject matter, which they do not, it remains unclear why one would have been motivated to combine them in a manner similar to this claim. Applicant can only conclude that the Office has impermissibly used the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious.

Accordingly, in view of the above discussion, the Office has not established a *prima facie* case of obviousness. Hence, for at least these reasons, this claim is allowable.

**Claims 18-27** depend from claim 16 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 16, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

Additionally, regarding claims 20-27, Applicant submits that the excerpts from Cohn that are cited by the Office disclose various classes, not interfaces. Accordingly, the Office's reliance on these excerpts is misplaced.

**Claim 28** recites a computer system including one or more microprocessors and one or more software programs, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services

comprising services related to constructing user interfaces, wherein the application program interface groups API functions into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics.

In making out the rejection of this claim, the Office relies on the same argument that it made with respect to claim 1.

Applicant respectfully disagrees and traverses the Office's rejection. First, as discussed above, Applicant submits that it is unclear what teachings of Cohn the Office is relying on for disclosing "an application program interface" (API), as claimed. Nevertheless, Applicant has thoroughly reviewed Chapters 5, 6, and 19 and is unable to find any discussion of an API, as recited in this claim. Instead, Chapter 5 describes how to use Java components with respect to creating a user interface and Chapter 6 describes how to combine and position these components. Furthermore, Chapter 17 merely discusses the nature of URLs and how they can be used. Missing from these excerpts, and from Cohn in general, is any discussion of an application programming interface that includes "separate commands to request services comprising services related to constructing user interfaces, wherein the application program interface groups API functions into multiple namespaces that define a collection of classes", as claimed.

Additionally, Applicant respectfully submits that the Office is interpreting the term "application program interface" in a manner that is inconstant with how it

is understood and used in the subject application. In this regard, Applicant directs the Office's attention to Page 6 (lines 8-10) of the subject application, which is one example of how this term is understood and used in the context of the subject application. Even a cursory inspection of Page 6 (lines 8-10) is sufficient to distinguish this claim from any teachings found in the cited excerpts of Cohn.

Furthermore, Applicant respectfully submits that the cited excerpts of Flanagan neither disclose nor suggest an API comprising all of the various types, as claimed. For instance, Applicant fails to see how Fig. 19-1 and Page 238 of Flanagan disclose "interfaces that define a contract that other types can implement", as claimed. Instead, this figure and excerpt merely discuss *classes* of the java.awt package.

Finally, and perhaps most importantly, as noted above, the Office's stated motivation merely describes the references themselves and fails to explain *why* one would have been motivated to combine their teachings. As such, this statement actually fails to articulate any motivation at all. Accordingly, as best as Applicant can discern, the Office's argument relies solely on the fact that the references *can* be combined. However, whether or not the references *can* be combined is irrelevant. Further, even if the cited references did disclose or suggest all the claimed subject matter, which they do not, it remains unclear why one would have been motivated to combine them in a manner similar to this claim. Applicant can only conclude that the Office has impermissibly used the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious.

Accordingly, in view of the above discussion, the Office has not established a *prima facie* case of obviousness. Hence, for at least these reasons, this claim is allowable.

**Claim 29** recites a method comprising:

- managing network and computing resources for a distributed computing system; and
- exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising functions to facilitate construction of user interfaces, wherein the functions are grouped into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics.

In making out the rejection of this claim, the Office simply indicates "see rejection of claim 1 above."

Applicant respectfully disagrees and traverses the Office's rejection. First, Applicant submits that it is unclear what teachings of Cohn the Office is relying on for disclosing "exposing a set of functions ... the set of functions comprising functions to facilitate construction of user interfaces", as claimed. Nevertheless, Applicant has thoroughly reviewed Chapters 5, 6, and 19 and is unable to find any discussion of exposing, as recited in this claim. Instead, Chapter 5 describes how to use Java components with respect to creating a user interface and Chapter 6 describes how to combine and position these components. Furthermore, Chapter 17 merely discusses the nature of URLs and how they can be used.

Furthermore, Applicant respectfully submits that the cited excerpts of Flanagan neither disclose nor suggest functions "grouped into multiple namespaces", as claimed. For instance, Applicant fails to see how Fig. 19-1 and Page 238 of Flanagan disclose "interfaces that define a contract that other types can implement", as claimed. Instead, this figure and excerpt merely discuss *classes* of the java.awt package.

Finally, and perhaps most importantly, as noted above, the Office's stated motivation merely describes the references themselves and fails to explain *why* one would have been motivated to combine their teachings. As such, this statement actually fails to articulate any motivation at all. Accordingly, as best as Applicant can discern, the Office's argument relies solely on the fact that the references *can* be combined. However, whether or not the references *can* be combined is irrelevant. Further, even if the cited references did disclose or suggest all the claimed subject matter, which they do not, it remains unclear why one would have been motivated to combine them in a manner similar to this claim. Applicant can only conclude that the Office has impermissibly used the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious.

Accordingly, in view of the above discussion, the Office has not established a *prima facie* case of obviousness. Hence, for at least these reasons, this claim is allowable.

**Claim 30** depends from claim 29 and is allowable as depending from an allowable base claim. This claim is also allowable for its own recited features which, in combination with those recited in claim 30, are neither disclosed nor

suggested in the references of record, either singly or in combination with one another.

**Claim 31** recites a method, comprising creating a namespace with functions that enable drawing and construction of user interfaces, the name space defining classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, structures that represent static allocated data that has value assignment semantics, and enumerations which are value types that represent named constants.

In making out the rejection of this claim, the Office simply indicates "see rejection of claim 5 above."

Applicant respectfully disagrees and traverses the Office's rejection. First, Applicant submits that it is unclear what teachings of Cohn the Office is relying on for disclosing "creating a namespace with functions that enable drawing and construction of user interfaces", as claimed. Nevertheless, Applicant has thoroughly reviewed Chapters 5, 6, and 19 and is unable to find any discussion of this subject matter.

Furthermore, Applicant respectfully submits that the cited excerpts of Flanagan neither disclose nor suggest a namespace with the functions claimed here. For instance, Applicant fails to see how Fig. 19-1 and Page 238 of Flanagan disclose "interfaces that define a contract that other types can implement", as claimed. Instead, this figure and excerpt merely discuss *classes* of the java.awt package.

Finally, and perhaps most importantly, as noted above, the Office's stated motivation merely describes the references themselves and fails to explain *why* one would have been motivated to combine their teachings. As such, this statement actually fails to articulate any motivation at all. Accordingly, as best as Applicant can discern, the Office's argument relies solely on the fact that the references *can* be combined. However, whether or not the references *can* be combined is irrelevant. Further, even if the cited references did disclose or suggest all the claimed subject matter, which they do not, it remains unclear why one would have been motivated to combine them in a manner similar to this claim. Applicant can only conclude that the Office has impermissibly used the claimed invention as an instruction manual or 'template' to piece together the teachings of the prior art so that the claimed invention is rendered obvious.

Accordingly, in view of the above discussion, the Office has not established a *prima facie* case of obviousness. Hence, for at least these reasons, this claim is allowable.

**Claims 32-40** depend from claim 31 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 31, are neither disclosed nor suggested in the references of record, either singly or in combination with one another.

Additionally, regarding claims 34-40, Applicant submits that the excerpts from Cohn that are cited by the Office disclose various classes, not interfaces. Accordingly, the Office's reliance on these excerpts is misplaced.
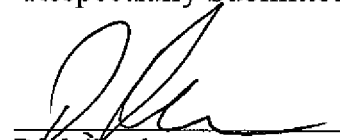
## Conclusion

The Office has failed to establish a *prima facie* case of obviousness. Accordingly, Applicant respectfully requests that the rejections be overturned and that the pending claims be allowed to issue.

Respectfully Submitted,

Dated: 09/07/2006    By: _____

Rich Bucher
Lee & Hayes, PLLC
Reg. No. 57,971
(509) 324-9256 ext. 216

## (8)    Appendix of Appealed Claims

1.    (Previously Presented)  A software architecture for a distributed computing system comprising:

an application configured to handle requests submitted by remote devices over a network; and

an application program interface to present functions used by the application to access network and computing resources of the distributed computing system, the application program interface comprising various types related to constructing user interfaces, wherein the various types comprise:

classes which represent managed heap allocated data that has reference assignment semantics;

interfaces that define a contract that other types can implement;

delegates that are object oriented function pointers;

structures that represent static allocated data that has value assignment semantics; and

enumerations which are value types that represent named constants.


3.    (Original)  A software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one client device.

4. (Original) A software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one server device that is configured as a Web server.

5. (Previously Presented) An application program interface embodied on one or more computer readable media, comprising: multiple types related to constructing user interfaces, the types comprising classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, structures that represent static allocated data that has value assignment semantics and enumerations which are value types that represent named constants.

6. (Original) An application program interface as recited in claim 5, wherein the classes comprise a forms class that represents a window or a dialog box that makes up an application's user interface.

7. (Original) An application program interface as recited in claim 6, wherein the forms class has multiple members comprising one or more of: public static properties, public static methods, public instance constructors, public instance methods, public instance properties, public instance events, protected instance properties, and protected instance methods.

8.    (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a button control interface that allows a control to act like a button on a form.

9.    (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a container control interface that provides functionality for a control to act as a parent for other controls.

10.    (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises an editing notification interface.

11.    (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a data object interface that provides a format independent mechanism for transferring data.

12.    (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a feature support interface that specifies a standard interface for retrieving feature information from a current system.

13.    (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a message filter interface.

14. (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises a handle-exposing interface to expose handles.

15. (Original) An application program interface as recited in claim 5, wherein the type comprising the interfaces comprises one or more of the following interfaces:

a button control interface that allows a control to act like a button on a form;

a container control interface that provides functionality for a control to act as a parent for other controls;

an editing notification interface;

a data object interface that provides a format independent mechanism for transferring data;

a feature support interface that specifies a standard interface for retrieving feature information from a current system;

a message filter interface; and

a handle-exposing interface to expose handles.

16. (Previously Presented) A distributed computer software architecture, comprising:

one or more applications configured to be executed on one or more computing devices, the applications handling requests submitted from remote computing devices;

33

a networking platform to support the one or more applications; and

an application programming interface to interface the one or more

applications with the networking platform, the application programming interface

comprising various types related to constructing user interfaces, wherein the

various types comprise:

classes which represent managed heap allocated data that has

reference assignment semantics;

interfaces that define a contract that other types can implement;

delegates that are object oriented function pointers;

structures that represent static allocated data that has value

assignment semantics; and

enumerations which are value types that represent named constants.


18. (Previously Presented) A distributed computer software architecture

as recited in claim 16, wherein the classes comprises a forms class that represents

a window or a dialog box that makes up an application's user interface.


19. (Original) A distributed computer software architecture as recited in

claim 18, wherein the forms class has multiple members comprising one or more

of: public static properties, public static methods, public instance constructors,

public instance methods, public instance properties, public instance events,

protected instance properties, and protected instance methods.

20.     (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises a button control interface that allows a control to act like a button on a form.

21.     (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises a container control interface that provides functionality for a control to act as a parent for other controls.

22.     (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises an editing notification interface.

23.     (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises a data object interface that provides a format independent mechanism for transferring data.

24.     (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises a feature support interface that specifies a standard interface for retrieving feature information from a current system.

25.    (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises a message filter interface.

26.    (Previously Presented)  A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises a handle-exposing interface to expose handles.

27.    (Previously Presented A distributed computer software architecture as recited in claim 16, wherein the type comprising the interfaces comprises one or more of the following interfaces:

a button control interface that allows a control to act like a button on a form;

a container control interface that provides functionality for a control to act as a parent for other controls;

an editing notification interface;

a data object interface that provides a format independent mechanism for transferring data;

a feature support interface that specifies a standard interface for retrieving feature information from a current system;

a message filter interface; and

a handle-exposing interface to expose handles.

28. (Previously Presented) A computer system including one or more microprocessors and one or more software programs, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services comprising services related to constructing user interfaces, wherein the application program interface groups API functions into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics.

29. (Previously Presented) A method, comprising:

managing network and computing resources for a distributed computing system; and

exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising functions to facilitate construction of user interfaces, wherein the functions are grouped into multiple namespaces that define a collection of classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, enumerations which are value types that represent named constants and structures that represent static allocated data that has value assignment semantics.

30. (Original) A method as recited in claim 29, further comprising receiving a request from a remote computing device, the request containing a call to the set of functions.

31. (Previously Presented) A method, comprising creating a namespace with functions that enable drawing and construction of user interfaces, the name space defining classes which represent managed heap allocated data that has reference assignment semantics, interfaces that define a contract that other types can implement, delegates that are object oriented function pointers, structures that represent static allocated data that has value assignment semantics, and enumerations which are value types that represent named constants.

32. (Original) A method as recited in claim 31, wherein the namespace defines a forms class that represents a window or a dialog box that makes up an application's user interface.

33. (Original) A method as recited in claim 32, wherein the forms class has multiple members comprising one or more of: public static properties, public static methods, public instance constructors, public instance methods, public instance properties, public instance events, protected instance properties, and protected instance methods.

34. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising a button control interface that allows a control to act like a button on a form.

35. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising a container control interface that provides functionality for a control to act as a parent for other controls.

36. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising an editing notification interface.

37. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising a data object interface that provides a format independent mechanism for transferring data.

38. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising a feature support interface that specifies a standard interface for retrieving feature information from a current system.

39. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising a message filter interface.

40. (Original) A method as recited in claim 31, wherein the namespace defines an interface comprising a handle-exposing interface to expose handles.

**(9)**     **Evidence appendix.  None**

**(10)    Related proceedings appendix.  None**